Lab #2 Answers

2020-01-24

Contents

Instructions **1**

Exercises from Chapter 3 **2**
Worked Example for One-Layer Atmosphere 2
Exercise 3.1 (Grad. students only) 4
Exercise 3.2 6
Exercise 3.3 8

Part 2: **10**
Exercises with CO_2 Data from the Mauna Loa Observatory 10
Exercises with Global Temperature Data from NASA 12

Instructions

This lab will introduce working with climate data using R and RMarkdown.

There are two parts to this lab assignment:

1. Part 1 is more like traditional homework. It consists of several exercises from chapter 3 of *Understanding the Forecast*, working with layer models of the greenhouse effect:

   - **All students** do Chapter 3, exercises 2–3.
   - **Graduate students** should also do Chapter 3, exercise 1.

   For the exercises, use the following numbers:

   - I_{solar} = 1350 W/m^2
   - $\sigma = 5.67 \times 10^{-8}$
   - $\alpha = 0.30$
   - $\varepsilon = 1.0$

   For Part 1, you have a choice of either working it like traditional homework on paper or else doing it with RMarkdown, using an RMarkdown template in the repository for the assignment. If you do it in Rmarkdown, you will use the template file I provided and fill in the R code to do the calculations and then use RMarkdown to include the answers in the text of the document.

2. For part 2, you will practice using R to organize and analyze data on the climate. We will download temperature and CO_2 measurements from scientific archives of climate data and practice producing tables of measurements, making plots of the measurements, and analyzing the series of measurements to determine the trends over time (rates of increase or decrease).

Both parts of the lab are due by the beginning of class on Friday, Jan. 24. If you do the layer model exercises on paper, turn in your work at the beginning of class. If you do the exercises with RMarkdown, knit your document to PDF or Word format and push it to GitHub. For Part 2, you must knit your work into a PDF or Word document and push the result to GitHub.

Exercises from Chapter 3

For the following exercises, use the following numbers:

- I_{solar} = 1350 W/m^2
- $\sigma = 5.67 \times 10^{-8}$
- $\alpha = 0.30$
- $\varepsilon = 1.0$

```
I_solar = 1350
alpha = 0.30
sigma = 5.67E-8
epsilon = 1
```

Worked Example for One-Layer Atmosphere

A One-Layer Model.

```
make_layer_diagram(1)
```

a) Write the energy budgets for the atmospheric layer, for the ground, and for the Earth as a whole.

Answer: Start at the top, at the boundary to space, and work downward:

- At the boundary to space, $I_{1,\text{up}} = (1-\alpha)I_{\text{solar}}/4$.
- At the atmospheric layer, $I_{1,\text{up}} + I_{1,\text{down}} = I_{\text{ground,up}}$
- At the ground, $(1-\alpha)I_{\text{solar}} + I_{1,\text{down}} = I_{\text{ground,up}}$

We also know that

- $I_{1,\text{up}} = I_{1,\text{down}} = \varepsilon\sigma T_1^4$
- $I_{\text{ground,up}} = \sigma T_{\text{ground}}^4$

b) Manipulate the budget for the Earth as a whole to obtain the temperature T_1 of the atmospheric layer. Does this part of the exercise seem familiar in any way? Does the term ring any bells?

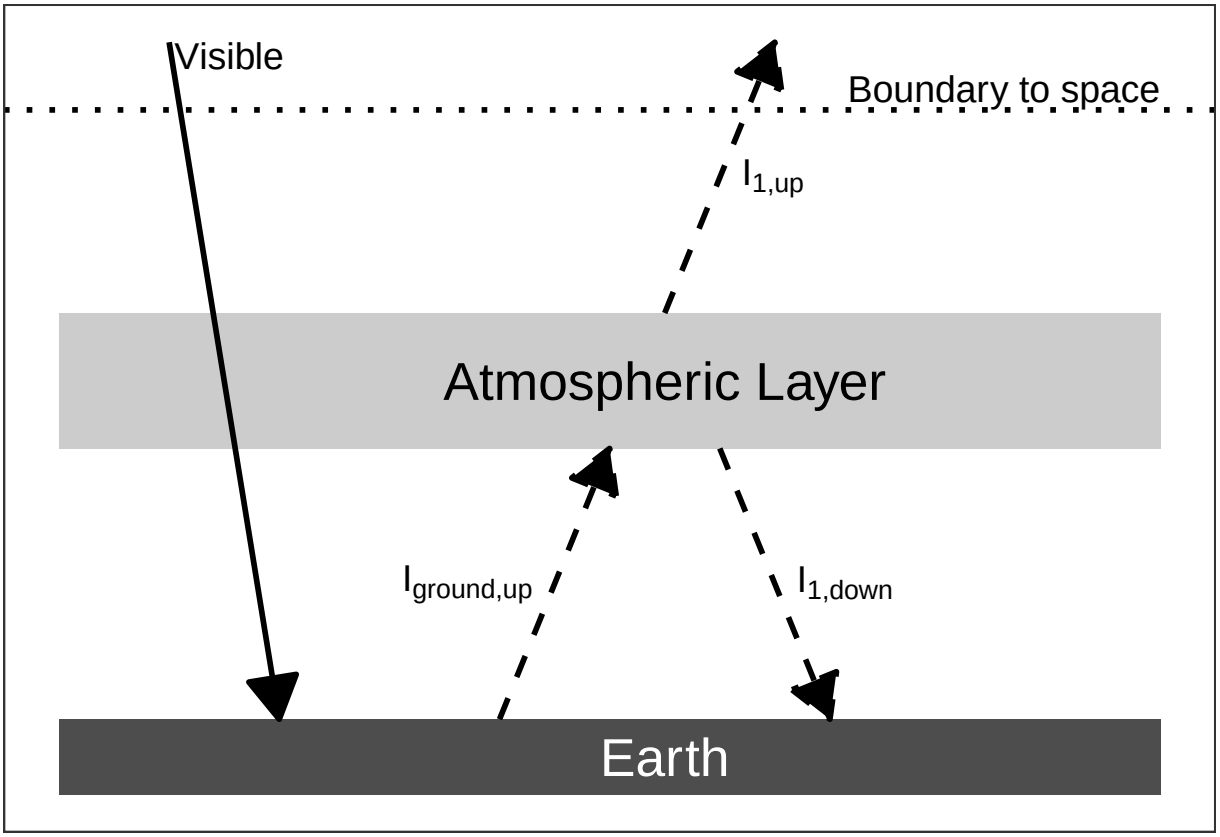


Figure 1: An energy diagram for a planet with one pane of glass for an atmosphere. The intensity of heat from visible light is $(1-\alpha)I_{\text{solar}}/4$.

Answer:

$$(1-\alpha)I_{\text{solar}}/4 = I_{1,\text{up}} = \sigma T_1^4$$

$$(1-\alpha)I_{\text{solar}}/4\varepsilon\sigma = T_1^4$$

$$T_1 = \sqrt[4]{\frac{(1-\alpha)I_{\text{solar}}}{4\varepsilon\sigma}}$$

This is familiar, because it's the same as the formula for the bare-rock temperature.

Here is R code to calculate $I_{1,\text{up}}$ and T_1:

```
I_1_up = (1 - alpha) * I_solar / 4
T_1 = (I_1_up / (epsilon * sigma))^0.25
```

From the algebraic solution, we expect T_1 to be 254. K. From the R code above, we get T_1 = 254. K.

c) Now insert the value you found for T_1 into the budget for atmospheric layer 1 to obtain the temperature of the ground, T_{ground}.

Answer:

- $I_{ground} = I_{1,up} + I_{1,down} = 2 \times I_{1,up}$
- $\varepsilon\sigma T_{ground}^4 = 2\varepsilon\sigma T_1^4$
- $T_{ground}^4 = 2T_1^4$
- $T_{ground} = \sqrt[4]{2} \times T_1$

And here is R code to calculate $I_{1,down}$, I_{ground}, and T_{ground}:

```
I_1_down = I_1_up
I_ground = I_1_up + I_1_down
T_ground = (I_ground / (epsilon * sigma))^0.25
```

From the algebraic solution, we get T_{ground} = 302. K and from the R code above, we get T_{ground} = 302. K.

Exercise 3.1 (Grad. students only)

The moon with no heat transport. The Layer Model assumes that the temperature of the body in space is all the same. This is not really very accurate, as you know that it is colder at the poles than it is at the equator. For a bare rock with no atmosphere or ocean, like the moon, the situation is even worse because fluids like air and water are how heat is carried around on the planet. So let us make the other extreme assumption, that there is no heat transport on a bare rock like the moon. Assume for comparability that the albedo of this world is 0.30, same as Earth's.

What is the equilibrium temperature of the surface of the moon, on the equator, at local noon, when the sun is directly overhead? (Hint: First figure out the intensity of the local solar radiation I_{solar})

Answer: Since the moon has no heat transport, we can consider I_{in} and I_{out} at the point where the sun is directly overhead, without worrying about averaging over all of the surface area of the moon. This means that instead of

$$I_{in} = \frac{(1-\alpha)I_{solar}}{4},$$

we use

$$I_{in} = (1-\alpha)I_{solar},$$

so

$$
\begin{aligned}
I_{\text{out}} &= I_{\text{in}} \\
\varepsilon\sigma T_{\text{moon}}^4 &= (1-\alpha) I_{\text{solar}} \\
T_{\text{moon}}^4 &= \frac{(1-\alpha) I_{\text{solar}}}{\varepsilon\sigma} \\
T_{\text{moon}} &= \sqrt[4]{\frac{(1-\alpha) I_{\text{solar}}}{\varepsilon\sigma}}
\end{aligned}
$$

```
I_solar = 1350 # W / m^2
albedo = 0.3
emissivity = 1
T_moon = ( (1 - albedo) * I_solar / (emissivity * sigma) )^0.25
```

T_{moon} is 359. Kelvin.

What is the equilibrium temperature on the dark side of the moon?

Answer:

On the dark side of the moon I_{solar} is zero, so we expect the temperature to be zero.

```
I_solar_dark = 0
T_dark_side = ( (1 - albedo) * I_solar_dark / (emissivity * sigma) )^0.25
```

$T_{\text{dark-side}}$ = 0 Kelvin.

In fact, outer space has a radiation that corresponds to black body with a temperature of around 3 Kelvin. This radiation is heat left over from the big bang, almost 14 billion years ago. Scientists call the the "Cosmic Microwave Background Radiation," and it means that empty deep space behaves as though it has a temperature of 3 K. Thus, in our example we would really expect the dark side of the moon to have a temperature of around 3 K.

The real dark side of the moon is cold, but not this cold.

- First, the moon orbits around the earth, so each side gets sunlight for half of each month and is in darkness for half of each month, so even when it's dark, the surface still has some leftover heat from the last time it was in the sun (it takes a long time to cool all the way from 359. K to 0 K).
- Second, there is a small, but nonzero, flow of heat through the solid moon (this is thermal conduction), so heat does travel from the bright side to the dark side.

These phenomena make the dark side of the moon much warmer than 3 Kelvin, but it is still bitterly cold there.

However, the homework explicitly told you to ignore these details, so for the purposes of this homework exercise, the correct answer is zero Kelvin.

Exercise 3.2

A Two-Layer Model. Insert another atmospheric layer into the model, just like the first one. The layer is transparent to visible light but a blackbody for infrared.

```
make_layer_diagram(2)
```

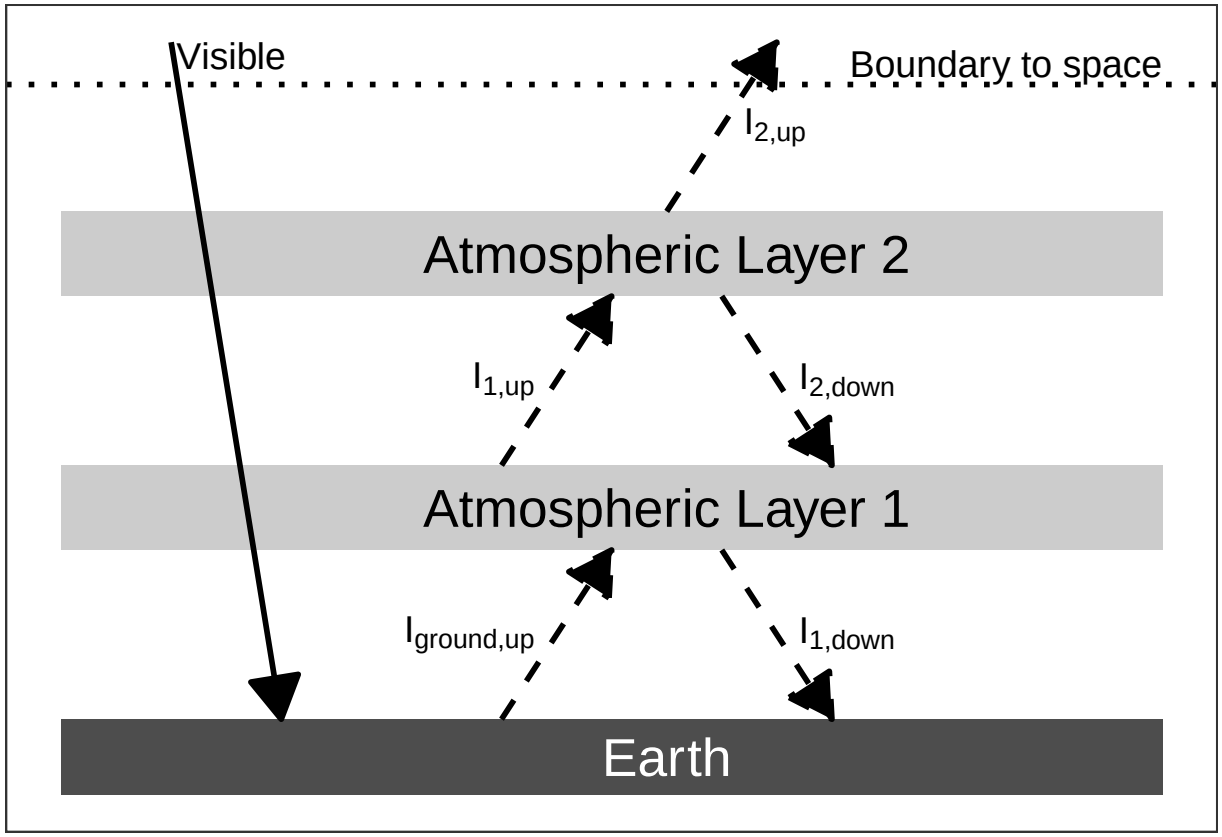


Figure 2: An energy diagram for a planet with two panes of glass for an atmosphere. The intensity of absorbed visible light is $(1-\alpha)I_{\text{solar}}/4$.

a) Write the energy budgets for both atmospheric layers, for the ground, and for the Earth as a whole, like we did for the One-Layer Model.

Answer:

First, we balance the flow of heat up and down at the imaginary boundary between the atmosphere and outer space.:

$$I_{2,\text{up}} = (1-\alpha)I_{\text{solar}}/4$$

The top layer of the atmosphere (layer 2) behaves exactly the way it did in the one-layer model:

$$I_{2,\text{in}} = I_{2,\text{out}}$$
$$I_{1,\text{up}} = I_{2,\text{up}} + I_{2,\text{down}} = 2I_{2,\text{up}}$$

($I_{2,\text{up}} = I_{2,\text{down}}$ because both are determined by the temperature of layer 2, T_2, and the Stefan-Boltzmann equation.)

This is essentially the same equation we used for the heat balance of the atmosphere in the one-layer model.

The heat-balance for the lower layer of the atmosphere is a little more complicated because there are two sources of heat in:

$$I_{1,\text{in}} = I_{1,\text{out}}$$
$$I_{\text{ground,up}} + I_{2,\text{down}} = I_{1,\text{up}} + I_{1,\text{down}} = 2I_{1,\text{up}}$$

Finally, the heat balance for the ground is

$$I_{\text{ground,in}} = I_{\text{ground,out}}$$
$$\frac{(1-\alpha)I_{\text{solar}}}{4} + I_{1,\text{down}} = I_{2,\text{up}} + I_{2,\text{down}} = 2I_{2,\text{up}}$$

b) Manipulate the budget for the Earth as a whole to obtain the temperature T_2 of the top atmospheric layer, labeled Atmospheric Layer 2 in the figure above. Does this part of the exercise seem familiar in any way? Does the term ring any bells?

Answer:

$$I_{2,\text{up}} = \frac{(1-\alpha)I_{\text{solar}}}{4}$$
$$\sigma T_2^4 = \frac{(1-\alpha)I_{\text{solar}}}{4}$$
$$T_2^4 = \frac{(1-\alpha)I_{\text{solar}}}{4\sigma}$$
$$T_2 = \sqrt[4]{\frac{(1-\alpha)I_{\text{solar}}}{4\sigma}}$$

```
T_2 = ( (1 - albedo) * I_solar / (4 * sigma) )^0.25
```

This is just like the one-layer model, and we get the same bare-rock temperature for the top of the atmosphere: T_2 = 254. Kelvin.

c) Insert the value you found for T_2 into the energy budget for layer 2, and solve for the temperature of layer 1 in terms of layer 2. How much bigger is T_1 than T_2?

Answer:

$$
\begin{aligned}
I_{2,\text{in}} &= I_{2,\text{out}} \\
I_{1,\text{up}} &= I_{2,\text{up}} + I_{2,\text{down}} = 2I_{2,\text{up}} \\
\sigma T_1^4 &= 2\sigma T_2^4 \\
T_1^4 &= 2T_2^4 \\
T_1 &= \sqrt[4]{2}\, T_2
\end{aligned}
$$

```
T_1 = 2^0.25 * T_2
```

This gives layer 1 the same temperature that the ground had in the one-layer model: T_1 = 302., which is $\sqrt[4]{2} = 1.19$ times bigger than T_2.

d) Now insert the value you found for T_1 into the budget for atmospheric layer 1 to obtain the temperature of the ground, T_{ground}. Is the greenhouse effect stronger or weaker because of the second layer?

Answer:

This gets a little more complicated:

$$
\begin{aligned}
I_{1,\text{in}} &= I_{1,\text{out}} \\
I_{\text{ground,up}} + I_{2,\text{down}} &= I_{1,\text{up}} + I_{1,\text{down}} = 2I_{1,\text{up}} \\
\sigma T_{\text{ground}}^4 + \sigma T_2^4 &= 2\sigma T_1^4 \\
T_{\text{ground}}^4 + T_2^4 &= 2T_1^4 \\
T_{\text{ground}}^4 &= 2T_1^4 - T_2^4
\end{aligned}
$$

But $T_1^4 = 2T_2^4$, so

$$
\begin{aligned}
T_{\text{ground}}^4 &= 4T_2^4 - T_2^4 = 3T_2^4 \\
T_{\text{ground}} &= \sqrt[4]{3}\, T_2^4
\end{aligned}
$$

```
T_ground = 3^0.25 * T_2
```

Thus, the ground temperature in the two-layer model is 334. Kelvin. This is 32. Kelvin hotter than the ground was in the one-layer model, so adding another layer made the greenhouse effect stronger.

Exercise 3.3

```
make_nuclear_winter_diagram()
```

Nuclear Winter. Let us go back to the One-Layer Model but change it so that the atmospheric layer absorbs visible light rather than allowing it to pass through (See the figure above). This could happen if the upper atmosphere were filled with dust. For simplicity, assume that the albedo of the Earth remains the same, even though in the real world it might change with a dusty atmosphere.> What is the temperature of the ground in this case?

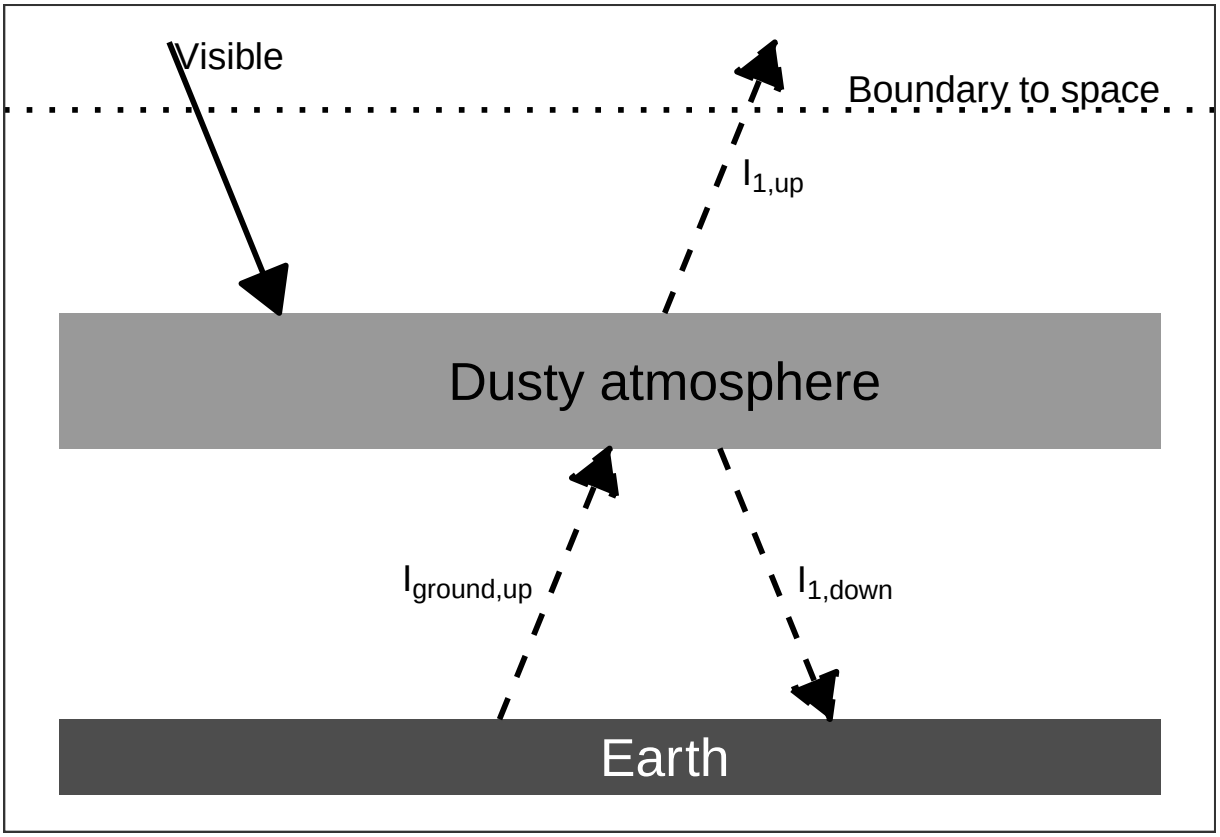


Figure 3: An energy diagram for a planet with an opaque pane of glass for an atmosphere. The intensity of absorbed visible light is $(1-\alpha)I_{\text{solar}}/4$.

Answer:

We start, as always, by balancing the energy. Just as in all of the other layer models, the temperature of the top layer of the atmosphere (the dusty layer) is the basic bare-rock temperature (because the dust doesn't change the albedo in this problem).

$$\begin{aligned} I_{\text{atm,up}} &= \frac{(1-\alpha)I_{\text{solar}}}{4} \\ \sigma T_1^4 &= \frac{(1-\alpha)I_{\text{solar}}}{4} \\ T_1^4 &= \frac{(1-\alpha)I_{\text{solar}}}{4\sigma} \\ T_1 &= \sqrt[4]{\frac{(1-\alpha)I_{\text{solar}}}{4\sigma}} \end{aligned}$$

What's new is that when we balance the heat flow in the dusty atmosphere, we have:

$$I_{\text{atm,in}} = I_{\text{atm,out}}$$

$$\frac{((1-\alpha)I_{\text{solar}}}{4} + I_{\text{ground,up}} = I_{\text{atm,up}} + I_{\text{atm,down}} = 2I_{\text{atm,up}}$$

We could just put numbers in and solve for T_{ground}, but if we remember that

$$\frac{(1-\alpha)I_{\text{solar}}}{4} = I_{\text{atm,up}}$$

we can substitute

$$\begin{aligned} I_{\text{atm,up}} + I_{\text{ground,up}} &= 2I_{\text{atm,up}} \\ I_{\text{ground,up}} &= I_{\text{atm,up}} \\ \sigma T_{\text{ground}}^4 &= \sigma T_{\text{atm}}^4 \\ T_{\text{ground}} &= T_{\text{atm}} \end{aligned}$$

```
T_atm = ( (1 - albedo) * I_solar / (4 * sigma) )^0.25
T_ground = T_atm
```

So the ground temperature will be 254. K, the same as the temperature of the atmosphere, which is the bare-rock temperature, so there is no greenhouse effect.

Part 2:

Exercises with CO_2 Data from the Mauna Loa Observatory

Using the `select` function, make a new data tibble called `mlo_seas`, from the original `mlo_data`, which only has two columns: `date` and `co2.seas`, where `co2.seas` is a renamed version of `co2.filled.seas` from the original tibble.

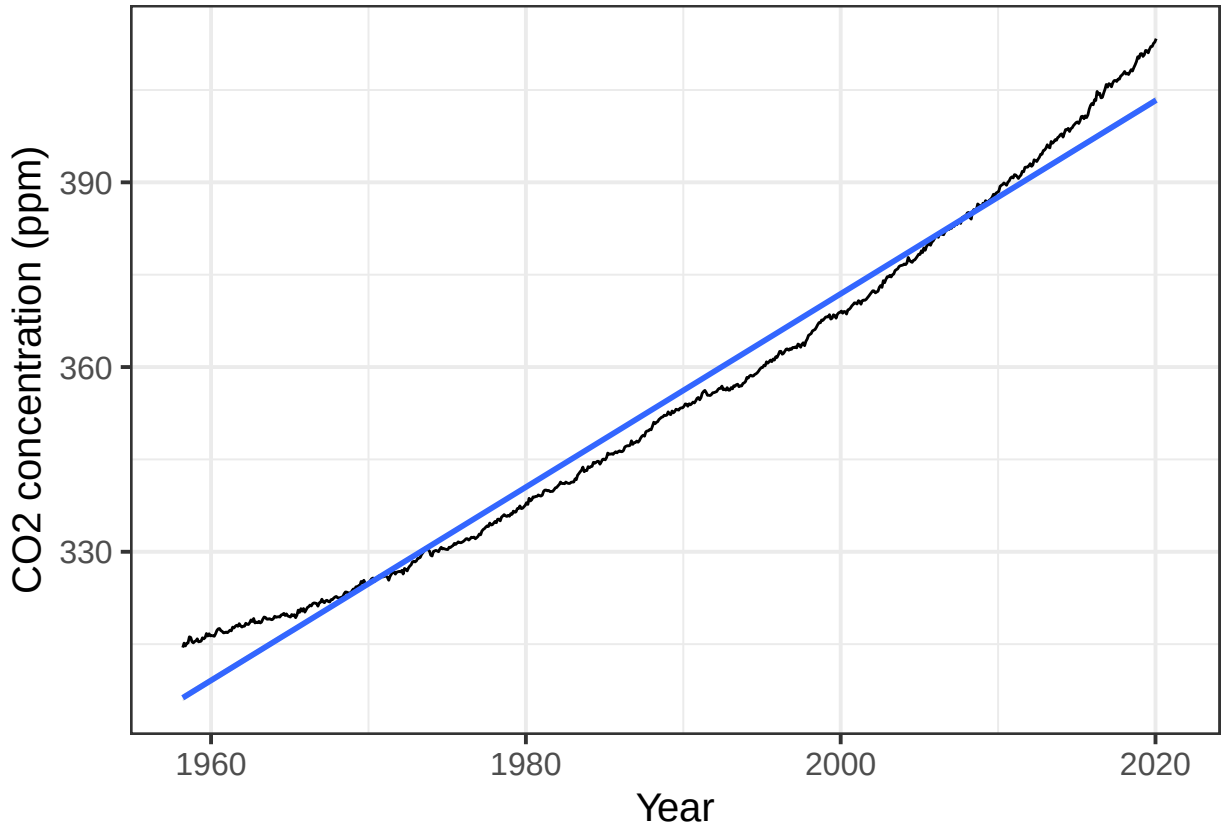
```
# We only need to load the libraries once and they will be loaded for all
# subsequent code chunks
library(tidyverse)
library(zoo)

mlo_seas = select(mlo_data, date, co2.filled.seas)
mlo_seas = rename(mlo_seas, co2.seas = co2.filled.seas)

# Alternately, you can simplify with the pipe operator:
#
# mlo_seas = select(mlo_data, date, co2.filled.seas) %>% rename(co2.seas = co2.filled.
#
# or you can rename as part of the select operation:
#
# mlo_seas = select(mlo_data, date, co2.seas = co2.filled.seas)
```

```
# Display the first few rows:
head(mlo_seas)
```

```
## # A tibble: 6 x 2
##    date co2.seas
##   <dbl>    <dbl>
## 1 1958.      NA
## 2 1958.      NA
## 3 1958.     314.
## 4 1958.     315.
## 5 1958.     315.
## 6 1958.     315.
```

Now plot this with `co2.seas` on the *y* axis and `date` on the *x* axis, and a linear fit:

```
ggplot(mlo_seas, aes(x = date, y = co2.seas)) +
  geom_line() +
  geom_smooth(method="lm") +
  labs(x = "Year", y = "CO2 concentration (ppm)")
```



Now fit a linear function to find the annual trend of `co2.seas`. Save the results of your fit in a variable called `trend.seas`.

```
trend.seas = lm(co2.seas ~ date, data = mlo_seas)

tidy(trend.seas)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept) -2767.    15.5         -179.       0
## 2 date            1.57   0.00779      201.       0
```

Compare the trend you fit to the raw `co2.filled` data to the trend you fit to the seasonally adjusted data.

Note: I just intend students to informally look at the trend in the graph and estimate its slope by eye to compare to the results in `trend.seas`.

Exercises with Global Temperature Data from NASA

We can also download a data set from NASA's Goddard Institute for Space Studies (GISS), which contains the average global temperature from 1880 through the present.

The URL for the data file is https://data.giss.nasa.gov/gistemp/tabledata_v4/GLB.Ts+dSST.csv

Download this file and save it in the directory `_data/global_temp_land_sea.csv`.

```
download.file(giss_url, file.path(data_dir, "global_temp_land_sea.csv"))
```

- Open the file in Excel or a text editor and look at it.
- Unlike the CO_2 data file, this one has a single line with the data column names, so you can specify `col_names=TRUE` in `read_csv` instead of having to write the column names manually.
- How many lines do you have to tell `read_csv` to skip?

Answer: 1 line: the first line is "Land-Ocean: Global Means" and we want to skip it.

- `read_csv` can automatically figure out the data types for each column, so you don't have to specify `col_types` when you call `read_csv`
- This file uses `***` to indicate missing values instead of `-99.99`, so you will need to specify `na="***"` in `read_csv`.

  For future reference, if you have a file that uses multiple different values to indicate missing values, you can give a vector of values to `na` in `read_csv`: `na = c("***","-99.99", "NA", "")` would tell `read_csv` that if it finds any of the values "***","-99.99","NA", or just a blank with nothing in it, any of those would correspond to a missing value, and should be indicated by `NA` in R.

Now read the file into R, using the `read_csv` function, and assign the resulting tibble to a variable `giss_temp`

```
giss_temp = read_csv(file.path(data_dir, "global_temp_land_sea.csv"),
                     skip = 1, na = "***", col_names = TRUE)

# show the first 5 lines of giss_temp
head(giss_temp, 5)
```

```
## # A tibble: 5 x 19
##    Year   Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov   Dec `J-D`
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1  1880 -0.17 -0.23 -0.08 -0.15 -0.09 -0.2  -0.17 -0.09 -0.13 -0.22 -0.21 -0.16 -0.16
## 2  1881 -0.19 -0.13  0.04  0.06  0.07 -0.18  0.01 -0.02 -0.14 -0.2  -0.17 -0.06 -0.08
## 3  1882  0.18  0.16  0.06 -0.15 -0.14 -0.21 -0.15 -0.06 -0.13 -0.23 -0.15 -0.35 -0.1
## 4  1883 -0.28 -0.35 -0.11 -0.18 -0.17 -0.06 -0.06 -0.13 -0.21 -0.11 -0.23 -0.1  -0.17
## 5  1884 -0.12 -0.08 -0.36 -0.39 -0.34 -0.34 -0.32 -0.27 -0.26 -0.24 -0.32 -0.3  -0.28
```

Something is funny here: Each row corresponds to a year, but there are columns for each month, and some extra columns called "J-D", "D-N", "DJF", "MAM", "JJA", and "SON". These stand for average values for the year from January through December, the year from the previous December through November, and the seasonal averages for Winter (December, January, and February), Spring (March, April, and May), Summer (June, July, and August), and Fall (September, October, and November).

The temperatures are recorded not as the thermometer reading, but as *anomalies*. If we want to compare how temperatures are changing in different seasons and at different parts of the world, raw temperature measurements are hard to work with because summer is hotter than winter and Texas is hotter than Alaska, so it becomes difficult to compare temperatures in August to temperatures in January, or temperatures in Texas to temperatures in Alaska and tell whether there was warming.

To make it easier and more reliable to compare temperatures at different times and places, we define anomalies: The temperature anomaly is the difference between the temperature recorded at a certain location during a certain month and a baseline reference value, which is the average temperature for that month and location over a period that is typically 30 years.

The GISS temperature data uses a baseline reference period of 1951–1980, so for instance, the temperature anomaly for Nashville in July 2017 would be the monthly average temperature measured in Nashville during July 2017 minus the average of all July temperatures measured in Nashville from 1951–1980.

The GISS temperature data file then averages the temperature anomalies over all the temperature-measuring stations around the world and reports a global average anomaly for every month from January 1880 through the latest measurements available (currently, November 2019).

Let's focus on the months only. Use `select` to select just the columns for "Year" and January through December (if you are selecting a consecutive range of columns between "Foo" and "Bar", you can call `select(Foo:Bar)`). Save the result in a variable called `giss_monthly`

```
giss_monthly = select(giss_temp, Year:Dec)
#
```

```
# alternately, you could remove unwanted columns:
#
# giss_monthly = select(giss_temp, -(`J-D`:SON))
#
# You have to use back-quotes for the column `J-D` because its name includes
# characters other than "a"-"z", "A"-"Z", "0"-"9", ".", and "_".
# You can give columns names with other characters than these, but it becomes
# more complicated to indicate them to R.

head(giss_monthly)
```

```
## # A tibble: 6 x 13
##    Year    Jan   Feb   Mar   Apr   May   Jun   Jul    Aug   Sep   Oct   Nov   Dec
##   <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>
## 1  1880 -0.17  -0.23 -0.08 -0.15 -0.09 -0.2  -0.17 -0.09  -0.13 -0.22 -0.21 -0.16
## 2  1881 -0.19  -0.13  0.04  0.06  0.07 -0.18  0.01 -0.02  -0.14 -0.2  -0.17 -0.06
## 3  1882  0.18   0.16  0.06 -0.15 -0.14 -0.21 -0.15 -0.06  -0.13 -0.23 -0.15 -0.35
## 4  1883 -0.28  -0.35 -0.11 -0.18 -0.17 -0.06 -0.06 -0.13  -0.21 -0.11 -0.23 -0.1
## 5  1884 -0.12  -0.08 -0.36 -0.39 -0.34 -0.34 -0.32 -0.27  -0.26 -0.24 -0.32 -0.3
## 6  1885 -0.580 -0.32 -0.25 -0.41 -0.44 -0.42 -0.32 -0.290 -0.27 -0.22 -0.22 -0.08
```

Next, it will be difficult to plot all of the data if the months are organized as columns. What we want is to transform the data tibble into one with three columns: "year", "month", and "anomaly". We can do this easily using the `gather` function from the `tidyverse` package: `gather(df, key = month, value = anomaly, -Year)` or `df %>% gather(key = month, value = anomaly, -Year)` will gather all of the columns except `Year` (the minus sign in `select` or `gather` means to include all columns except the ones indicated with a minus sign) and:

- Make a new tibble with three columns: "Year", "month", and "anomaly"
- For each row in the original tibble, make rows in the new tibble for each of the columns "Jan" through "Dec", putting the name of the column in "month" and the anomaly in "anomaly".

Here is an example of using `gather`, using the built-in data set `presidents`, which lists the quarterly approval ratings for U.S. presidents from 1945–1974:

```
df = presidents@.Data %>% matrix(ncol=4, byrow = TRUE) %>%
  as_tibble() %>% set_names(paste0("Q", 1:4)) %>% mutate(year = 1944 + seq(n()))


print("First 10 rows of df are")
```

```
## [1] "First 10 rows of df are"
```

```
print(head(df, 10))
```

```
## # A tibble: 10 x 5
##       Q1    Q2    Q3    Q4  year
```

```
##    <dbl> <dbl> <dbl> <dbl> <dbl>
##  1    NA    87    82    75  1945
##  2    63    50    43    32  1946
##  3    35    60    54    55  1947
##  4    36    39    NA    NA  1948
##  5    69    57    57    51  1949
##  6    45    37    46    39  1950
##  7    36    24    32    23  1951
##  8    25    32    NA    32  1952
##  9    59    74    75    60  1953
## 10    71    61    71    57  1954
```

For each year, the table has a column for the year and four columns (Q1 ... Q4) that hold the quarterly approval ratings for the president in that quarter. Now we want to gather these data into three columns: one column for the year, one column to indicate the quarter, and one column to indicate the approval rating.

We do this with the `gather` function from the `tidyverse` package.

```
dfg <- df %>% gather(key = quarter, # create a column called "quarter" to store
                                    # the names of the columns that are gathered
                     value = approval, # create a column called "approval" to
                                       # store the values from those columns
                                       # (i.e., the approval ratings in that
                                       # quarter)
                     -year # the minus sign means gather all columns EXCEPT year.
                     ) %>%
  arrange(year, quarter) # sort the rows of the resulting tibble to put
                         # the years in ascending order, from 1945 to 1971
                         # and within each year, sort the quarters from Q1
                         # to Q4

head(dfg) # print the first few rows of the tibble.
```

```
## # A tibble: 6 x 3
##    year quarter approval
##   <dbl> <chr>      <dbl>
## 1  1945 Q1            NA
## 2  1945 Q2            87
## 3  1945 Q3            82
## 4  1945 Q4            75
## 5  1946 Q1            63
## 6  1946 Q2            50
```

Now you try to do the same thing to:

- First select just the columns of `giss_monthly` for the year and the individual months.

- Next, gather all the months togeher, so there will be three columns: one for the year, one for the name of the month, and one for the temperature anomaly in that month.
- Store the result in a new variable called `giss_g`

```
giss_g = gather(giss_monthly, key = month, value = anomaly, -Year)
```

Remember how the CO_2 data had a column `date` that had a year plus a fraction that corresponded to the month, so June 1960 was 1960.4548?

Here is a trick that lets us do the same for the `giss_g` data set. R has a data type called `factor` that it uses for managing categorical data, such as male versus female, Democrat versus Republican, and so on. Categorical factors have a textual label, but are silently represented as integer numbers. Normal factors don't have a special order, so R sorts the values alphabetically. However, there is another kind of factor called an ordered factor, which allows us to specify the order of the values.

We can use a built-in R variable called `month.abb`, which is a vector of abbreviations for months.

The following command will convert the `month` column in `giss_g` into an ordered factor that uses the integer values 1, 2, ..., 12 to stand for "Jan", "Feb", ..., "Dec", and then uses those integer values to create a new column, `date` that holds the fractional year, just as the `date` column in `mlo_data` did:

```
giss_g = giss_g %>%
  mutate(month = ordered(month, levels = month.abb),
         date = Year + (as.integer(month) - 0.5) / 12) %>%
  arrange(date)`
```

In the code above, `ordered(month, levels = month.abb)` converts the variable `month` from a character (text) variable that contains the name of the month to an ordered factor that associates a number with each month name, such that "Jan" = 1 and "Dec" = 12.

Then we create a new column called `date` to get the fractional year corresponding to that month. We have to explicitly convert the ordered factor into a number using the function `as.integer()`, and we subtract 0.5 because the time that corresponds to the average temperature for the month is the middle of the month.

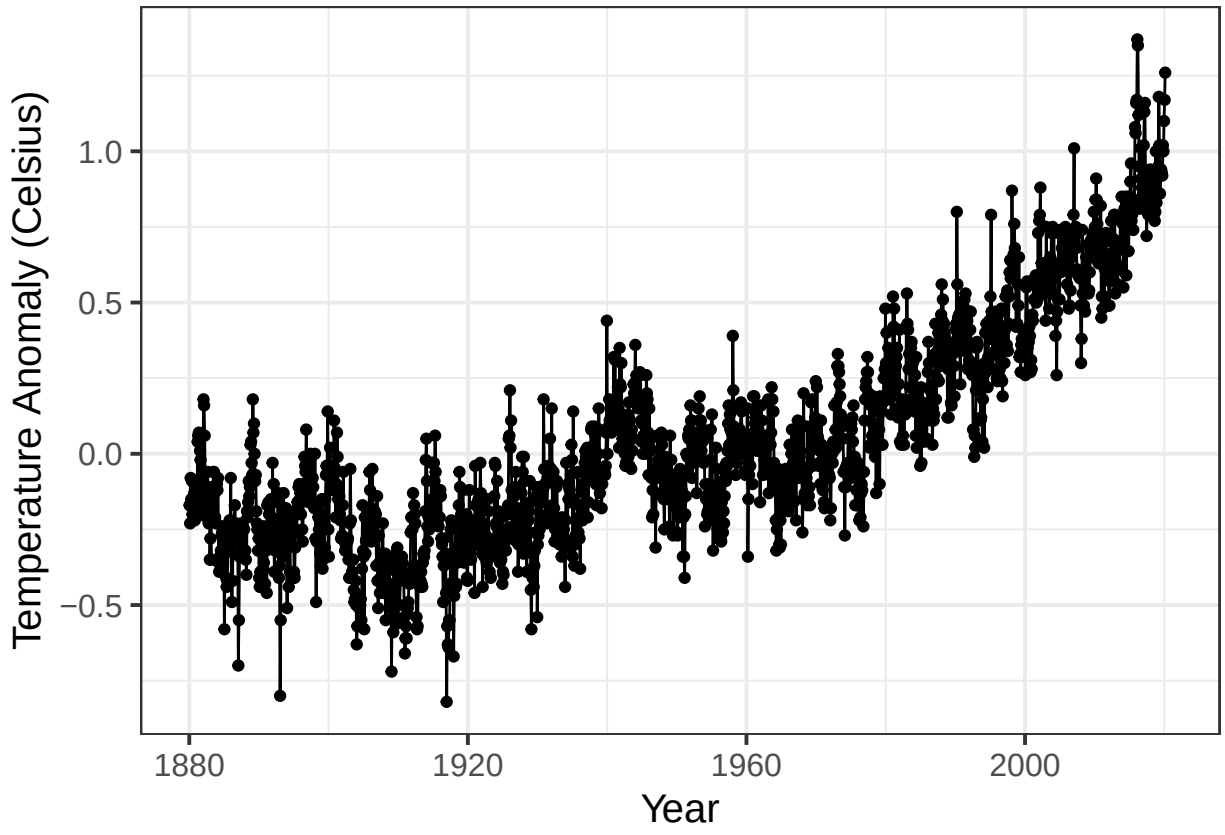
Below, use code similar to what I put above to add a new `date` column to `giss_g`.

```
# Here, you just copy the code from above and run it.
#
giss_g = giss_g %>%
  mutate(month = ordered(month, levels = month.abb),
         date = Year + (as.integer(month) - 0.5) / 12) %>%
  arrange(date)
```

Now plot the monthly temperature anomalies versus date:

```
ggplot(giss_g, aes(x = date, y = anomaly)) +
  geom_line() +
```

```
  geom_point() +
  labs(x = "Year", y = "Temperature Anomaly (Celsius)")
```

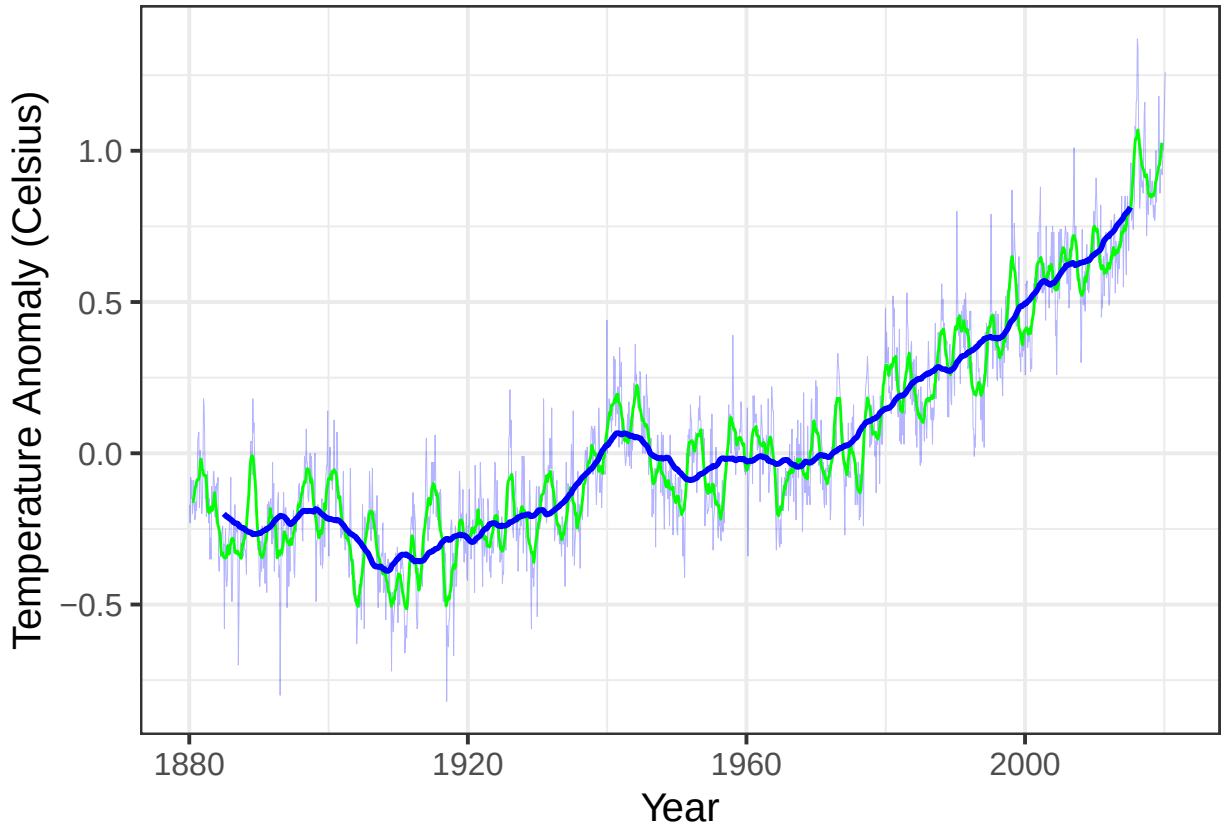


That plot probably doesn't look like much, because it's very noisy. Use the function `rollapply` from the package `zoo` to create new columns in `giss_g` with 12-month and 10-year (i.e., 120-month) rolling averages of the anomalies.

Make a new plot in which you plot a thin blue line for the monthly anomaly (use `geom_line(aes(y = anomaly), color = "blue", alpha = 0.3, size = 0.1)`; alpha is an optional specification for transparency where 0 means invisible (completely transparent) and 1 means opaque), a medium dark green line for the one-year rolling average, and a thick dark blue line for the ten-year rolling average.

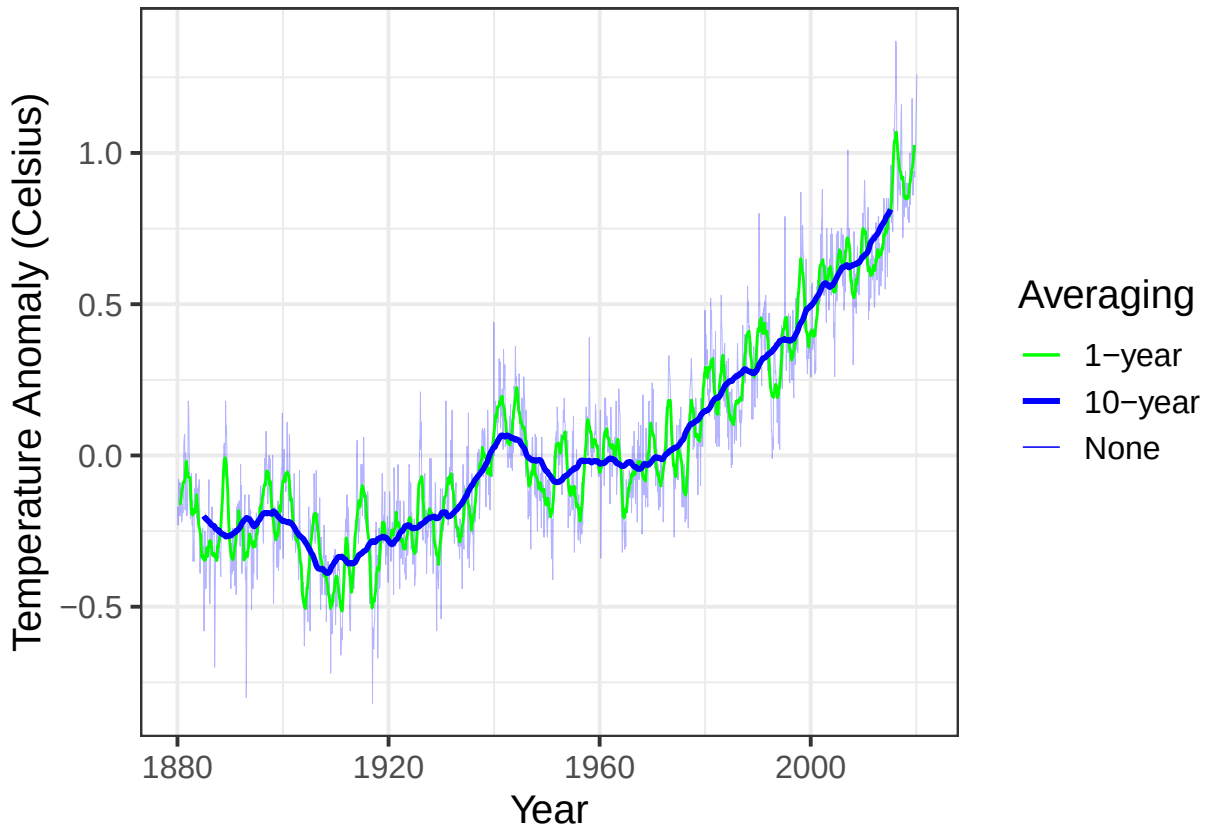
```
giss_g %>%
   mutate( smooth.1 = rollapply(data = anomaly, width = 12, FUN = mean,
                                fill = NA, align = "center"),
           smooth.10 = rollapply(data = anomaly, width = 120, FUN = mean,
                                 fill = NA, align = "center")) %>%
  ggplot(aes(x = date)) + # Put code here to map variables to aesthetics
  geom_line(aes(y = anomaly), alpha = 0.3, size = 0.1, color = "blue") +
  geom_line(aes(y = smooth.1), color = "green", size = 0.5) +
  geom_line(aes(y = smooth.10), color = "blue", size = 1) +
```

```
  labs(x = "Year", y = "Temperature Anomaly (Celsius)")
```



Alternately, we could do this fancier version:

```
giss_g %>%
   mutate( smooth.1 = rollapply(data = anomaly, width = 12, FUN = mean,
                                fill = NA, align = "center"),
           smooth.10 = rollapply(data = anomaly, width = 120, FUN = mean,
                                 fill = NA, align = "center")) %>%
  ggplot(aes(x = date)) + # Put code here to map variables to aesthetics
  geom_line(aes(y = anomaly, size = "None", color = "None"), alpha = 0.3) +
  geom_line(aes(y = smooth.1, size = "1-year", color = "1-year")) +
  geom_line(aes(y = smooth.10, size = "10-year", color = "10-year")) +
  scale_color_manual(values = c("None" = "blue", "1-year" = "green",
                                "10-year" = "blue"), name = "Averaging") +
  scale_size_manual(values = c("None" = 0.1, "1-year" = 0.5,
                               "10-year" = 1.0), name = "Averaging") +
  labs(x = "Year", y = "Temperature Anomaly (Celsius)")
```



The graph shows that temperature didn't show a steady trend until starting around 1970, so we want to isolate the data starting in 1970 and fit a linear trend to it.

To select only rows of a tibble that match a condition, we use the function `filter` from the `tidyverse` package:

`data_subset = df %>% filter( conditions )`, where `df` is your original tibble and `conditions` stands for whatever conditions you want to apply. You can make a simple condition using equalities or inequalities:

- `data_subset = df %>% filter( month == "Jan")` to select all rows where the month is "Jan"
- `data_subset = df %>% filter( month != "Aug")` to select all rows where the month is not August.
- `data_subset = df %>% filter( month %in% c("Sep", "Oct", "Nov")` to select all rows where the month is one of "Sep", "Oct", or "Nov".
- `data_subset = df %>% filter(year >= 1945)` to select all rows where the year is greater than or equal to 1945.
- `data_subset = df %>% filter(year >= 1951 & year <= 1980 )` to select all rows where the year is between 1951 and 1980, inclusive.

- `data_subset = df %>% filter(year >= 1951 | month == "Mar")` to select all rows where the year is greater than or equal to 1951 or the month is "Mar". this will give all rows from January 1951 onward, plus all rows before 1951 where the month is March.

Below, create a new variable `giss_recent` and assign it a subset of `giss_g` that has all the data from January 1970 through the present. Fit a linear trend to the monthly anomaly and report it.

What is the average change in temperature from one year to the next?

```
giss_recent = filter(giss_g, date >= 1970)

recent_trend = lm(anomaly ~ date, data = giss_recent)

tidy(recent_trend)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept) -37.5     0.802        -46.7 4.27e-202
## 2 date          0.0190  0.000402      47.2 2.34e-204
```

Did Global Warming Stop after 1998?

It is a common skeptic talking point that global warming stopped in 1998. In years with strong El Niños, global temperatures tend to be higher and in years with strong La Niñas, global temperatures tend to be lower. We will discuss why later in the semester.

The year 1998 had a particularly strong El Niño, and the year set a record for global temperature that was not exceeded for several years. Indeed, compared to 1998, it might look as though global warming paused for many years.

We will examine whether this apparent pause has scientific validity.

To begin with, we will take the monthly GISS temperature data and convert it to annual average temperatures, so we can deal with discrete years, rather than separate temperatures for each month.

We do this with the `group_by` and `summarize` functions.

We also want to select only recent data, so we arbitrarily say we will look at temperatures starting in 1979, which gives us 19 years before the 1998 El Ni~o.

We don't have a full year of data for 2017, so we want to discard that because we won't get a full year average from it.

If we go back to the original `giss_g` data tibble, run the following code:

```
giss_annual = giss_g %>%
  filter(Year >= 1979 & Year < 2017) %>%
  group_by(Year) %>%
  summarize(anomaly = mean(anomaly)) %>%
```

```
  ungroup() %>%
  mutate(date = Year + 0.5)

head(giss_annual)
```

```
## # A tibble: 6 x 3
##    Year anomaly  date
##   <dbl>   <dbl> <dbl>
## 1  1979   0.162 1980.
## 2  1980   0.257 1980.
## 3  1981   0.321 1982.
## 4  1982   0.137 1982.
## 5  1983   0.311 1984.
## 6  1984   0.155 1984.
```

This code groups the giss data by the year, so that one group will have January–December 1979, another will have January–December 1980, and so forth.

Then we replace the groups of 12 rows for each year (each row represents one month) with a single row that represents the average of those 12 months.

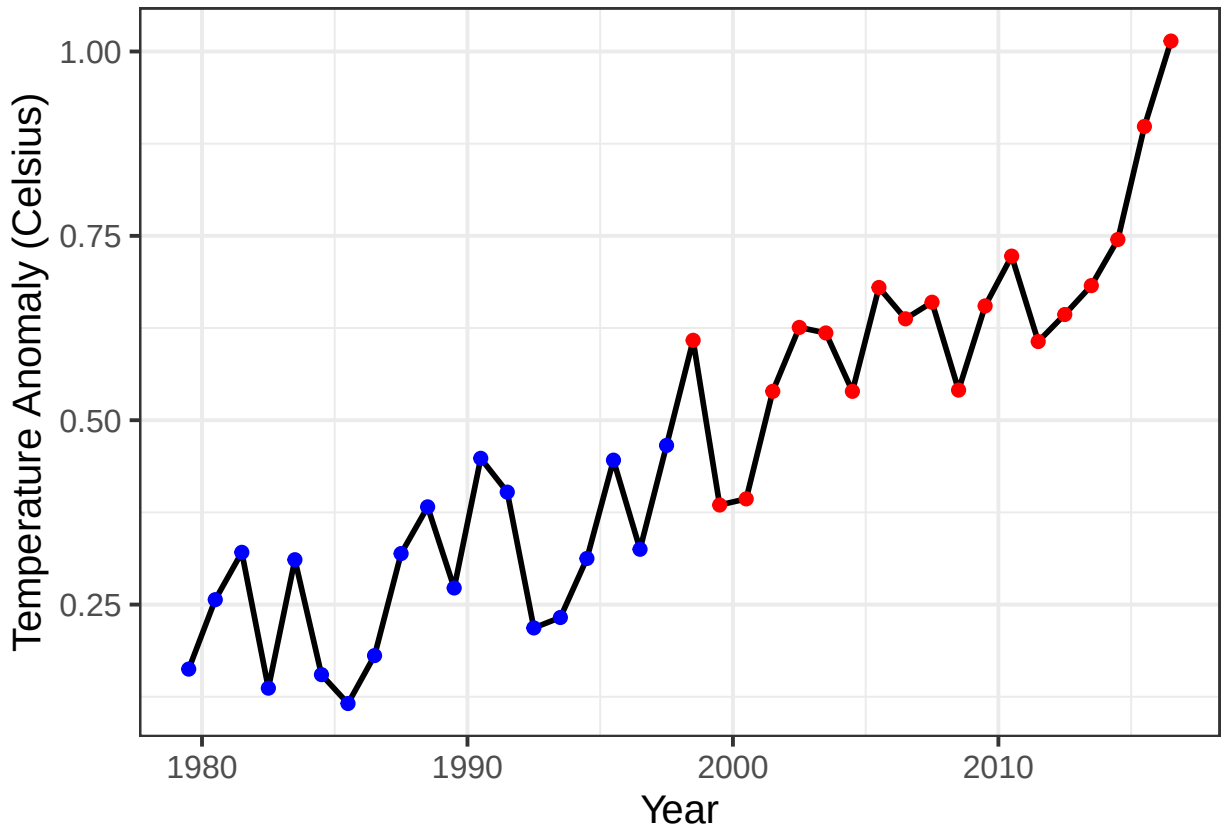
It is important to tell R to `ungroup` the data after we're done working with the groups.

Finally, we set `date` to `year + 0.5` because the average of a year corresponds to the middle of the year, not the beginning.

Now, let's introduce a new column `after`, which indicates whether the data is after the 1998 El Niño:

Now plot the data and color the points for 1998 and afterward dark red to help us compare before and after 1998.

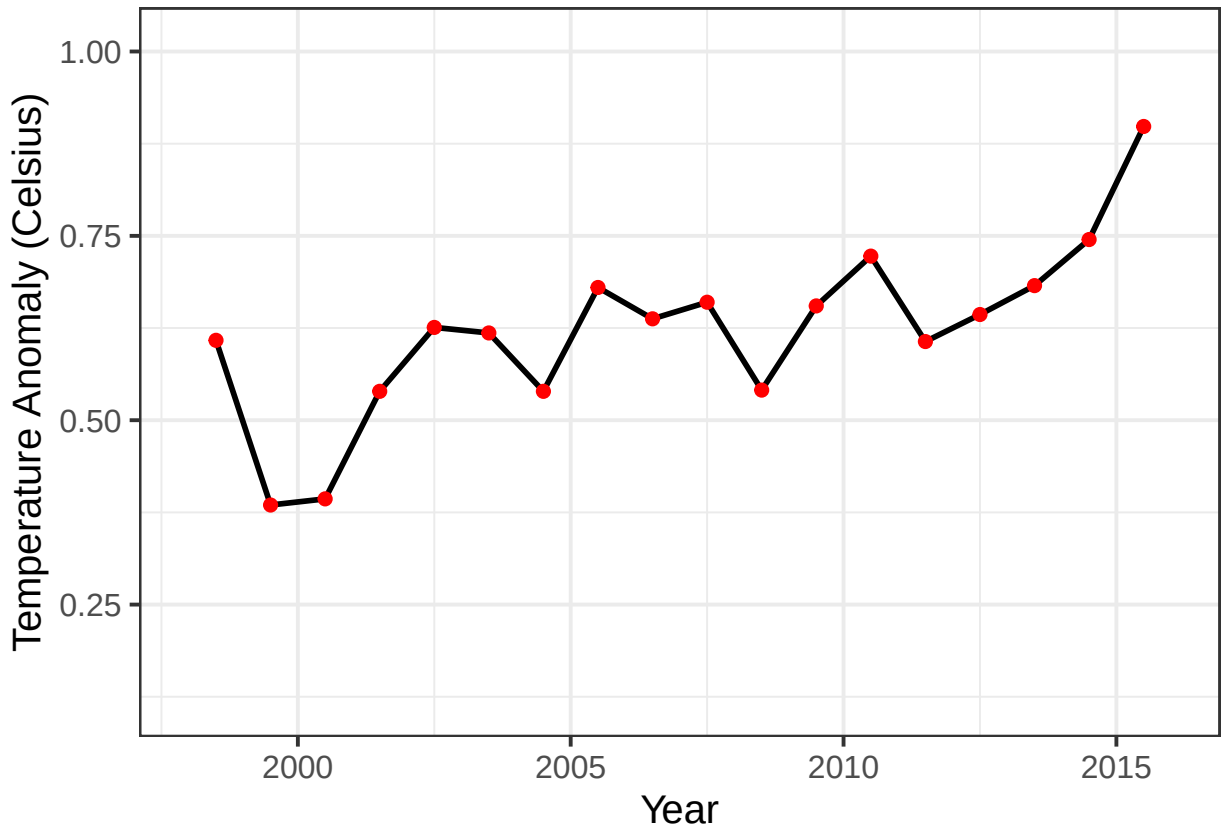
```
ggplot(giss_annual, aes(x = date, y = anomaly)) +
  geom_line(size = 1) +
  # I didn't include it in the original instructions, but the
  # following version of geom_line is nicer than what's above:
  #
  # geom_line(aes(color = Year >= 1998), size = 1) +
  #
  geom_point(aes(color = Year >= 1998), size = 2) +
  scale_color_manual(values = c("TRUE" = "red", "FALSE" = "blue"),
                     guide = "none") + # color "before" points blue,
                                       # "after" points red
                                       # don't use a legend
  labs(x = "Year", y = "Temperature Anomaly (Celsius)")
```



Does it look as though the red points are not rising as fast as the blue points?

Let's just plot the data from 1998 on:

```
ggplot(giss_annual, aes(x = date, y = anomaly)) +
  geom_line(size = 1) +
  # I didn't include it in the original instructions, but the
  # following version of geom_line is nicer than what's above:
  #
  # geom_line(aes(color = Year >= 1998), size = 1) +
  #
  geom_point(aes(color = Year >= 1998), size = 2) +
  scale_color_manual(values = c("TRUE" = "red", "FALSE" = "blue"),
                     guide = "none") + # color "before" points blue,
                                       # "after" points red
                                       # don't use a legend
  xlim(1998,2016) +
  labs(x = "Year", y = "Temperature Anomaly (Celsius)")
```



Now how does it look?

Let's use the `filter` function to break the data into two different tibbles: `giss_before` will have the data from 1979–1998 and the other, `giss_after` will have the data from 1998 onward (note that the year 1998 appears in both tibbles).

```
giss_before = filter(giss_annual, Year <= 1998)
giss_after = filter(giss_annual, Year >= 1998)
```

Now use `lm` to fit a linear trend to the temperature data in `giss_before` (from 1979–1998) and assign it to a variable `giss_trend`.

Next, add a column `timing` to each of the split data sets and set the value of this column to "Before" for `giss_before` and "After" for `giss_after`.

```
giss_before = mutate(giss_before, timing = "Before")
giss_after = mutate(giss_after, timing = "After")

giss_trend = lm(anomaly ~ date, data = giss_before)
tidy(giss_trend)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
```

```
## 1 (Intercept) -28.0      7.68        -3.65 0.00185
## 2 date          0.0142   0.00386      3.69 0.00169
```

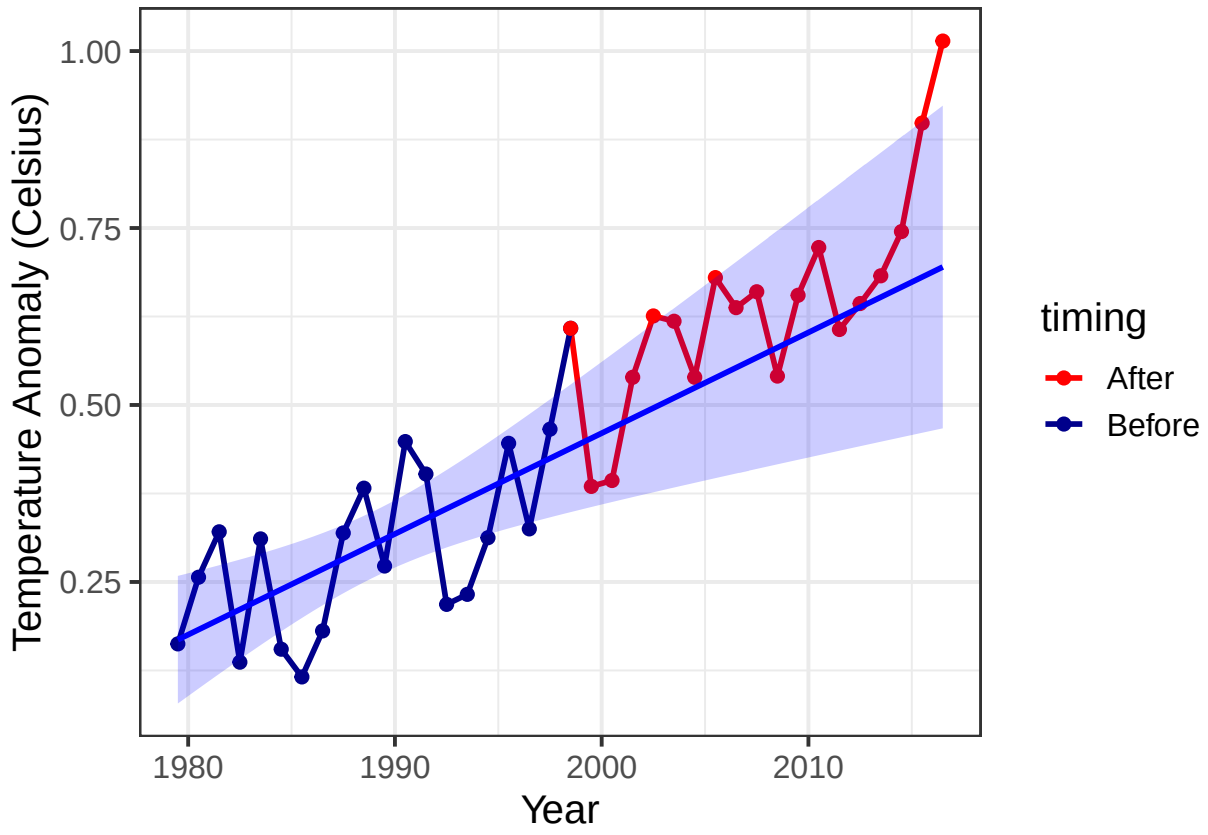
Now, combine the two tibbles into one tibble:

```
giss_combined <- bind_rows(giss_before, giss_after)
```

Now let's use ggplot to plot `giss_combined`:

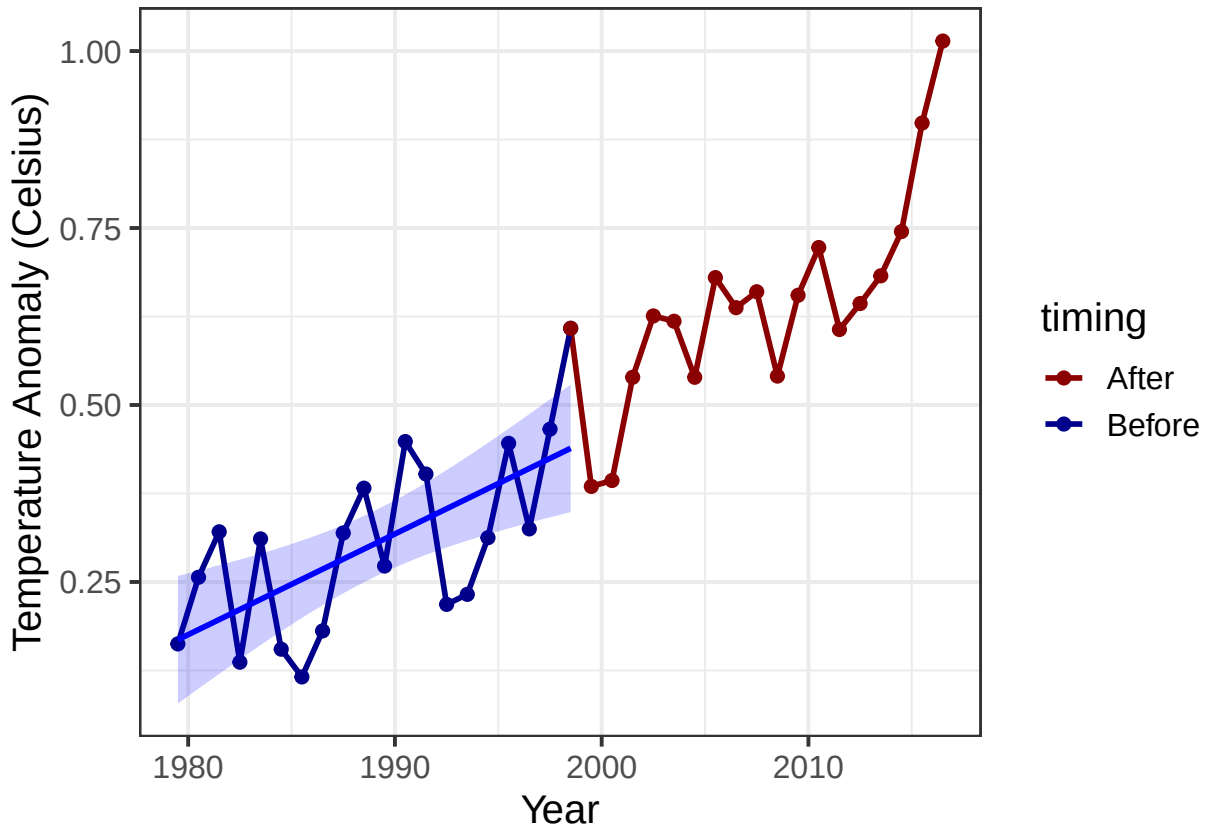
- Aesthetic mapping:
  - Use the `date` column for the *x* variable.
  - Use the `anomaly` column for the *y* variable.
  - Use the `timing` column to set the color of plot elements
- Plot both lines and points.
  - Set the `size` of the lines to 1
  - Set the `size` of the points to 2
- Use the `scale_color_manual` function to set the color of "Before" to "blue" and "After" to "red"
- Use `geom_smooth(data = giss_before, method="lm", color = "blue", fill = "blue", alpha = 0.2, fullrange = TRUE)` to show a linear trend that is fit just to the `giss_before` data.

```
ggplot(giss_combined, aes(x = date, y = anomaly, color = timing)) +
  geom_line(size = 1) +
  geom_point(size = 2) +
  geom_smooth(data = giss_before, method = "lm", color = "blue", fill = "blue",
              alpha = 0.2, fullrange = TRUE) +
  scale_color_manual(values = c(Before = "darkblue", After = "red")) +
  labs(x = "Year", y = "Temperature Anomaly (Celsius)")
```

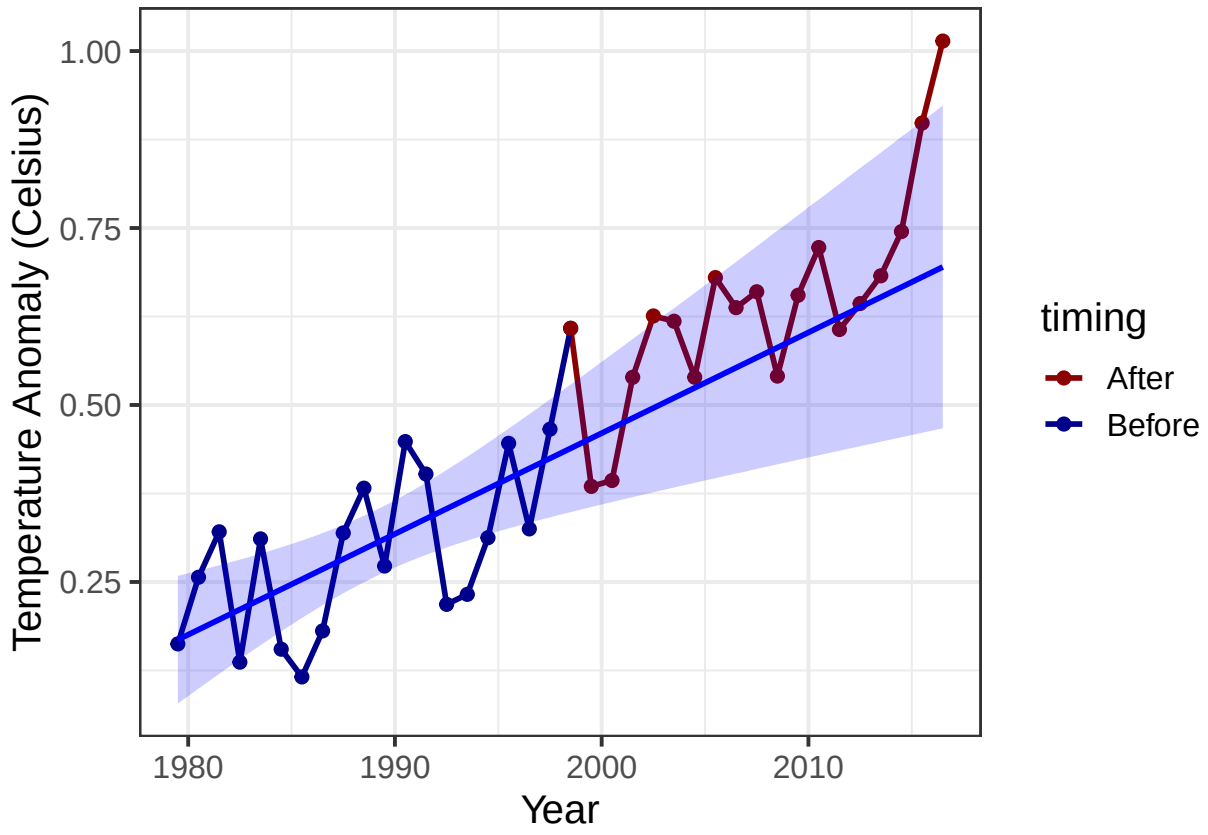


Try this with the parameter `fullrange` set to `TRUE` and `FALSE` in the `geom_smooth` function. What is the difference?

```
ggplot(giss_combined, aes(x = date, y = anomaly, color = timing)) +
  geom_line(size = 1) +
  geom_point(size = 2) +
  geom_smooth(data = giss_before, method = "lm", color = "blue", fill = "blue",
              alpha = 0.2, fullrange = FALSE) +
  scale_color_manual(values = c(Before = "darkblue", After = "darkred")) +
  labs(x = "Year", y = "Temperature Anomaly (Celsius)")
```




```
ggplot(giss_combined, aes(x = date, y = anomaly, color = timing)) +
  geom_line(size = 1) +
  geom_point(size = 2) +
  geom_smooth(data = giss_before, method = "lm", color = "blue", fill = "blue",
              alpha = 0.2, fullrange = TRUE) +
  scale_color_manual(values = c(Before = "darkblue", After = "darkred")) +
  labs(x = "Year", y = "Temperature Anomaly (Celsius)")
```



Answer: Both plots show the full data set, and a linear trend that is fit just to the "before" data. The trend line shows both the best fit for a trend (that's the solid line) and the range of uncertainty in the fit (that's the light blue shaded area around the line).

But, when `fullrange = FALSE`, the line is only drawn for the data to which the trend was fit, whereas when `fullrange = TRUE`, the trend line is drawn for the full range of the graph, even though the trend was only fit to the data in part of the graph.

If the temperature trend changed after 1998 (e.g., if the warming paused, or if it reversed and started cooling) then we would expect the temperature measurements after 1998 to fall predominantly below the extrapolated trend line, and our confidence that the trend had changed would depend on the number of points that fall below the shaded uncertainty range.

How many of the red points fall below the trend line?

Answer: 6 points: 1999, 2000, 2008, 2011, 2012, and 2013.

How many of the red points fall above the trend line?

Answer: Not counting 1998, 12 points: 2001, 2002, 2003, 2004 (barely), 2005, 2006, 2007, 2009, 2010, 2014, 2015, and 2016.

What do you conclude about whether global warming paused or stopped after 1998?

Answer: Most of the years after 1998 were warmer than we would have predicted if temperatures

had continued to follow the warming trends of 1979–1998, so this is not evidence of any slow-down or pause in the warming.